Fewer Flops at the Top: Accuracy, Diversity, and Regularization in Two-Class Collaborative Filtering

Paudel, Bibek ; Haas, Thilo ; Bernstein, Abraham

Abstract: In most existing recommender systems, implicit or explicit interactions are treated as positive links and all unknown interactions are treated as negative links. The goal is to suggest new links that will be perceived as positive by users. However, as signed social networks and newer content services become common, it is important to distinguish between positive and negative preferences. Even in existing applications, the cost of a negative recommendation could be high when people are looking for new jobs, friends, or places to live. In this work, we develop novel probabilistic latent factor models to recommend positive links and compare them with existing methods on five different openly available datasets. Our models are able to produce better ranking lists and are effective in the task of ranking positive links at the top, with fewer negative links (flops). Moreover, we find that modeling signed social networks and user preferences this way has the advantage of increasing the diversity of recommendations. We also investigate the effect of regularization on the quality of recommendations, a matter that has not received enough attention in the literature. We find that regularization parameter heavily affects the quality of recommendations in terms of both accuracy and diversity.

DOI: https://doi.org/10.1145/3109859.3109916

Other titles: I Want to Watch Non-Popcorn Movies Sometimes: Accuracy, Diversity, and Regularization in Probabilistic Latent Factor Models

Fewer Flops at the Top: Accuracy, Diversity, and Regularization in Two-Class Collaborative Filtering

Bibek Paudel
Department of Informatics
University of Zurich
Zurich, Switzerland
paudel@ifi.uzh.ch

Thilo Haas
University of Zurich
Zurich, Switzerland
thilo@thilohaas.ch

Abraham Bernstein
Department of Informatics
University of Zurich
Zurich, Switzerland
bernstein@ifi.uzh.ch

ABSTRACT

In most existing recommender systems, implicit or explicit interactions are treated as positive links and all unknown interactions are treated as negative links. The goal is to suggest new links that will be perceived as positive by users. However, as signed social networks and newer content services become common, it is important to distinguish positive from negative preferences. Even in existing applications, the cost of negative recommendations could be high when people are looking for new jobs, friends, or places to live.

In this work, we develop novel probabilistic latent factor models to recommend positive links and compare them with existing methods on five different openly available datasets. Our models are able to produce better ranking lists and are effective in the task of ranking positive links at the top, with fewer negative links (flops). Moreover, we find that modeling signed social networks and user preferences this way has the advantage of increasing the diversity of recommendations. We also investigate the effect of regularization on the quality of recommendations, a matter that has not received enough attention in the literature. We find that regularization parameter heavily affects the quality of recommendations in terms of both accuracy and diversity.

CCS CONCEPTS

•**Information systems** →**Recommender systems;** *Information extraction;* Document filtering; •**Human-centered computing** →*Social networks;*

KEYWORDS

matrix factorization, positive and negative recommendations, collaborative filtering, diverse recommendations
DOI: http://dx.doi.org/10.1145/3109859.3109916

1 INTRODUCTION

Collaborative Filtering has found use in a wide range of applications from movies to book recommendation, and from job suggestion to matchmaking. In essence, these models produce personalized ranking lists for users such that the choices perceived as positive by the user are ranked towards the top of her list. They do so by differentiating previous *positive* choices from *all other* choices of the user. In other words, user preference is modeled as a binary variable: known positive preferences are treated as positive, while unknown or negative preferences are treated as negative preferences. The personalized ranking thus generated for the user is only concerned with the benefit of putting positive items at the top of the list. Since both negative and unknown items are clubbed together, these systems treat the cost of them appearing at the top of the list as the same.

This assumption may be flawed in many cases. Consider a choice that costs more than a few minutes of time wasted by listening to a bad song. Jobs, education, cities to live, or friends to make are some choices that require a user to be more invested. In such cases, it is undesirable to recommend negative items high in the ranked list, since the user would not appreciate them. Unknown and negative items cannot be treated as having the same cost in this setting. We would like to have a system that puts choices perceived as positive by the user at the top of her ranking list, while at the same time having as few negative choices (flops) as possible at the top of that list. This is the focus of our work in this paper.

Increasingly, negative feedback is becoming more common in the setting of both explicit and implicit feedback. Examples of explicit negative feedback are buttons like *dislike* and *downvote* on popular online services, whereas skipping to the next song within few seconds of listening to it can be considered an example of implicit negative feedback. In the latter example, skipping a song may not necessarily imply dislike, but rather an act of trying to find a familiar song. The problem of interpreting implicit signals is tricky and needs careful problem-specific consideration.

Negative and positive links also feature in other domains, most notably social networks. In case of signed social networks, two nodes can have a relationship that is labeled as positive or negative. Examples of positive relations may include friendship or trust, and mistrust maybe an example of negative relation. People choose to follow or friend other people in social networks, indicating positive preference. In other cases, they choose to block, unfriend or mute, as an expression of negative preference.

Naturally, as in item-recommender systems, a link recommendation service has to be able to differentiate these important distinctions. Failing to separate negative preferences from positive ones may make the service unattractive to the user, reducing its usability and ultimately efficiency. Again, the goal – as in recommender systems – is to rank other people in the social network in a way that positive links appear at the top, with fewer negative links.

In this work, we propose novel techniques to achieve the goal stated above. Specifically, our main contributions are: (a) we define

RecSys '17, August 27–31, 2017, Como, Italy.

DOI: http://dx.doi.org/10.1145/3109859.3109916

the problem of Two-Class Collaborative Filtering (TCCF) (b) we develop new probabilistic latent factor models to deal with the TCCF problem: TC-MF (Two-Class Matrix Factorization) and its variation TC-MF1, and S-MF (SoftMax Matrix Factorization), (c) we show with results in five different datasets and in comparison with other state-of-the-art methods that our approach is able to generate rankings with positive items at the top and negative at the bottom.

We find that modeling signed social networks and user preferences this way has advantages other than ranking accuracy: our recommendations are more diverse whilst consistently being among the best in terms of recommendation accuracy, a quality desirable in recommender systems. We also investigate the effect of regularization parameter in the quality of recommendations, a matter that has not received enough attention in the literature. We find that regularization parameter significantly affects both accuracy and the diversity of recommendations.

2 PROBLEM DEFINITION

Our work is general enough to be applied to any domain with pairwise (dyadic) interaction between nodes. This includes the settings of recommender systems and social network link prediction. Graphs and matrices lend themselves as general frameworks to model such data and relationships.

Consider a set of nodes V and a set of directed edges between them E, forming a graph $G(V, E)$. Depending on the dataset, G could be bi-bipartite, in which case the set of left-vertices and right-vertices are M and N respectively. In the case of a user-item graph, M denotes the set of users and N the set of items and $V = M \cup N$. For example, in a movie recommendation system, N could be the set of movies available in the system. In such cases, the edge set is $E \subseteq M \times N$. In the case of a social network, there is only one type of node: the users of the network and the edge set is: $E \subseteq V \times V$. For simplicity, in both cases, we'll refer to the graph as user-item graph or *interaction graph*.

When the graph is not bi-bipartite, the set of nodes that have at least one incoming edges is treated as N and the set of nodes that have at least one outgoing edge is treated as M. Although these are not disjoint sets, from the view of each node in M, all the other nodes can be considered as items and from the view of each node in N, all other nodes can be considered as users. The cardinality of these sets are denoted as $|M|$ and $|N|$. A single edge is denoted as an ordered pair $(i, j) \in E$ such that $i \in M, j \in N$, or simply as $e_{i,j}$. Additionally, each edge in the graph has a label among $L_{obs} = \{+1, -1\}$. All pairs in $\{M \times N\} \setminus E$ (absent/possible edges) have the label 0 (or ?) indicating unknown label. The complete set of labels is then: $L = L_{obs} \cup \{0\} = \{+1, -1, 0\}$. The label of an edge can be denoted in functional form as: $f_L : V \times V \rightarrow L$. Although the datasets we use will only have these three categorical labels, the term *rating* will be used interchangeably with labels. Relationships between nodes are also referred to as interactions, feedbacks, or links. Links are called positive links or negative links based on their labels. The feedback dataset set can also be modeled as a matrix: $\mathbf{P} \in \{+1, -1, 0\}^{|M| \times |N|}$ and defined as: $p_{i,j} = f_L(e_{i,j})$.

The matrix $\mathbf{P}$ is called the user-item matrix or simply *interaction/feedback matrix*. A subset of the edges from E are separated as training set E_{tr}, correspondingly a training matrix $\mathbf{R}$ can be defined with entries $r_{i,j}$. Based on $\mathbf{R}$, our goal is then to learn a model that ranks the nodes in N for each node in M, with the goal that potentially negative links are at the bottom and potentially positive links at the top of the ranking.

This is the **Two-Class Collaborative Filtering Problem**.

One way of measuring such a ranking is using the AUC measure. For any given user (in M), consider the training set as: $(n_i, l_i)_{i=1}^{n}$ where $n_i \in N$ and $l_i \in L_{obs}$. Let $\mathcal{P} = \{n_i \mid l_i = 1\}$ be the set of positive training samples and $\mathcal{N} = \{n_i \mid l_i = -1\}$ be the set of negative samples. AUC is then defined as:

$$AUC = \frac{1}{|\mathcal{P}|\,|\mathcal{N}|} \sum_{n_i \in \mathcal{P}} \sum_{n_j \in \mathcal{N}} \delta(f(n_i) > f(n_j)) \tag{1}$$

where $\delta(f)$ is an indicator function which is 1 if f is satisfied and 0 otherwise. The AUC in (1) measures how many positive items are ranked above negative items and normalizes it by the total number of possible pairs. Its value is 1 for a perfect ranking (i.e., each positive item is ranked above all negative items) and 0.5 for a random ranking. AUC is the value of the Wilcoxon-Mann-Whitney statistic [7]. Note that this AUC is slightly different when used to rank only positive items- in that case it suffices to rank positive items above all the other items.

Another measure, proposed in [19] is called the Generalized AUC (GAUC). Unlike AUC in (1), GAUC also considers the relative ranking of positive items and negative items with respect to the class of unknown items. Specifically, it is highest when all positive items are ranked at the top, all unknown items are ranked in the middle and the negative items are ranked at the bottom. Similar to above, consider the training set $(n_i, l_i)_{i=1}^{n}$ with l_i now defined to include unknown labels: $l_i \in L$. The set of samples with unknown labels is $\mathcal{O} = \{n_i \mid l_i = 0\}$. Then GAUC is:

$$\begin{aligned} GAUC = \frac{1}{|\mathcal{P}| + |\mathcal{N}|}\Big(\frac{1}{|\mathcal{O}| + |\mathcal{N}|} \sum_{n_i \in \mathcal{P}} \sum_{n_s \in \mathcal{O} \cup \mathcal{N}} \delta(f(n_i) \\ > f(n_s)) + \frac{1}{|\mathcal{O}| + |\mathcal{P}|} \sum_{n_j \in \mathcal{N}} \sum_{n_t \in \mathcal{O} \cup \mathcal{P}} \delta(f(n_j) > f(n_t))\Big) \end{aligned} \tag{2}$$

3 RELATED WORK

Broadly, our work in this paper can be compared with previous work in the domain of recommender systems and social network link recommendation using matrix factorization. We discuss previous work from these areas in this section.

Collaborative Filtering [2, 6, 16] was used early on in recommendation systems. Most of these systems deal with explicit feedback datasets. These include numeric rating (usually 1-5) provided by a user to express her preference to the items. Such methods try to model the dataset as completely as possible. The missing values in the interaction matrix are filled using user's mean rating or global mean rating. The aim is to learn a model from a partially observed dataset that is close to the original rating matrix.

One-Class Collaborative Filtering (OCCF) was introduced by [13]. OCCF relaxed the need to model the data completely and to differentiate between different kinds of preferences. In OCCF, separating known positive preferences from unknown or negative preferences is enough because the goal is to recommend links of positive class only. In this way, incomplete matrices with 1 and ? can be used to model the input dataset and predict positive links.

Modeling interaction data as a binary matrix in this way has found widespread use and popularity. OCCF forms the backbone of many online services that suggest next movie to watch or a new restaurant to try. Similar systems are also used for link recommendation in social networks.

Matrix Factorization techniques aim to find two low-dimensional factor matrices to approximate and complete the missing values in **R** based on some loss function. Their application in Recommender Systems has been reviewed in [11]. Probabilistic Matrix Factorization [10, 17, 18] has shown to outperform vanilla Matrix Factorization. Bayesian Personalized Ranking [15] (BPRMF) is another matrix factorization method that aims to maximize the pairwise ranking between positive and negative items. This translates to maximizing the AUC (Area Under the ROC curve) and thus fits more naturally to the ranking task than the usually adopted approach of minimizing the loss over the positive items alone.

Similar to [8], Logistic Matrix Factorization [10] factorizes the observed user-item rating matrix **R** into two low dimensional matrices **A** and **B**, but using a logistic loss instead of squared loss.

We argue that it is often equally important to distinguish negative links from positive ones. Our focus in this work is to rank positive links at the top and negative links at the bottom for each user's ranked list.

GAUC-OPT [19] is a recently introduced model that aims to do so by optimizing a pairwise ranking measure called the Generalized AUC (GAUC) as in (2). Our work is motivated by the problem introduced in [19], who in turn extended the work by [15]. Previously, [13] considered a different but related problem of balancing the extent of treating missing values as negative examples. They don't try to distinguish between missing and negative examples, but treat all missing values as negative examples. Their focus is to balance positive and negative examples by introducing several weighting schemes. The Matrix Factorization techniques introduced above employ models similar to [13], with the weighing of the implicit feedbacks. Different from our work, in [15], the focus is on differentiating positive links from all other links; the optimization depends on sampling triplets from positive and unknown preferences, whereas we use different latent factor formulations.

In this work, we focus on probabilistic latent factor models in order to solve the problem of the Two Class Collaborative Filtering. Our models are distinctively different from these approaches. Unlike the models discussed above, we explicitly model the probability of positive and negative link in our likelihood function, enabling the model to better differentiate between those two classes. We use a probabilistic approach, i.e., estimate the probability of positive and negative feedback and use it for ranking. Ranking links this way automatically places positive items at the top and negative at the bottom, meaning that items with unknown preferences will appear in the middle. This is not the case with other existing latent factor models. Our results show that this modeling choice is indeed beneficial and improves performance compared to other models.

4 TWO CLASS COLLABORATIVE FILTERING

Latent factor models decompose the training matrix $\mathbf{R}^{|M|\times|N|}$ into two low dimensional factor matrices $\mathbf{A}^{|M|\times k}$ and $\mathbf{B}^{k\times|N|}$ with $k < |M|$, $k < |N|$, such that their product approximates the original matrix **R** according to some loss function.

In other words, if $E(.)$ is some loss function, then the goal is to minimize $E(\mathbf{R}, \mathbf{AB}) + \lambda(\mathbf{A}, \mathbf{B})$, where $\lambda(\mathbf{A}, \mathbf{B})$ is regularization term to prevent overfitting. Typically, squared loss is used, which allows for quick analytical solution in the form of Alternating Least Squares (ALS) by fixing one factor at a time [8].

ALS based methods have proved to be useful in both tasks: rating prediction and ranking. Since we are dealing with more than one class in this work, we take a probabilistic approach that helps us directly model the probability of a user liking or disliking a link. In that, we opt for a logistic loss function as a natural choice instead of a squared loss function. This means that we can't use ALS based optimization, but have to move towards the local minima using a gradient based approach.

Matrix Factorization with logistic loss has been described in [18] and recently in [10]. In this scenario, the probability that a user interacts with an item is modeled according to the logistic function as follows:

$$p(r^+_{u,i} \mid a_u, b_i, \beta_u, \beta_i) = \frac{\exp(a_u b_i^T + \beta_u + \beta_i)}{1 + \exp(a_u b_i^T + \beta_u, +\beta_i)} \tag{3}$$

The terms β_u and β_i are bias factors associated with each user u and item i, and account for the differences among users and items respectively. The vectors a_u's and b_i's are rows and columns of the factor matrices **A** and **B**.

Under the assumption that all observed ratings $r_{u,i}$ in **R** are independent, it optimizes the following likelihood function, where α defines a weighting factor to balance between positive and unknown ratings as suggested by [8]:

$$p^{LK} = \prod_{(u,i)\in M\times N} p(r^+_{u,i} \mid a_u, b_i, \beta_u, \beta_i)^{\alpha r_{u,i}} (1 - p(r^+_{u,i} \mid a_u, b_i, \beta_u, \beta_i)) \tag{4}$$

To distinguish between unknown and negative feedback, we explicitly account for negative classes within the likelihood function and introduce three models: i) Two-Class Matrix Factorization 1 (TC-MF1), ii) Two-Class Matrix Factorization (TC-MF), and iii) Softmax Matrix Factorization (S-MF). While TC-MF1 and TC-MF are adaptations of the existing probabilistic matrix factorization methods, S-MF introduces a novel way to approach the problem. In S-MF, we use two separate latent features for each item, corresponding to positive and negative preferences. Thus S-MF needs to learn twice as many parameters per item compared to the two other models.

Next we describe our new models and formulate their optimization schemes using the log likelihood function according to [9], which are able to directly capture the user's preference towards positive and negative items.

4.1 TC-MF1

The goal of TC-MF1 is first to differentiate between positive and negative or unknown feedback and second to use the resulting matrix factorization to provide a probability measure for classifying positive and negative feedback. To distinguish between the positive and negative items, we denote the probability that a user u likes link i as $p(r^+_{ui} \mid a_u, b_i, \beta_u, \beta_i)$ and the probability that u dislikes link k as: $p(r^-_{uk} \mid a_u, b_k, \beta_u, \beta_k) = 1 - p(r^+_{uk} \mid a_u, b_k, \beta_u, \beta_k)$. For better readability we omit the conditionals in the remainder of this paper,

defining the probabilities simply as $p(r^+_{ui})$ and $p(r^-_{uk})$. We also use $\mathcal{F}$ to represent the matrix factorization results, $\mathcal{F} := a_u b_i^T + \beta_u + \beta_i$

This model differs from [10] in the definition of the likelihood function. We only consider the probability of a negative rating for truly negative or unknown samples within the likelihood function, instead of for every rating being considered. We therefore only account for the loss of false positives, whereas [10] also accounts for the loss of true positives. With this adaption we expect the model to perform better in arranging truly positive samples on top of a user's ranking list.

With f_L as defined in Section 2, $\mathcal{F}$ as defined above, and δ as an indicator function, we consider the following likelihood function:

$$p^{LK} = \prod_{(u,i)\in M\times N} p(r^+_{ui})^{\delta(f_L(u,i)=+1)}(1-p(r^+_{ui}))^{1-\delta(f_L(u,i)=+1)}$$

The resulting log-likelihood function is:

$$\log p^{LK} = \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\mathcal{F} - \log(1+\exp(\mathcal{F}))$$

4.2 TC-MF

In this approach, we use a different likelihood function to account for the probability of a negative rating only if the given user rated this item negative. We include positive and negative feedback and ignore unknown feedback within the likelihood function.

This is different from the objective for [10] described above: we are explicitly modeling the probability of the user's preference of being positive or negative in our likelihood function, whereas the former model only assigns different weights to the probability of a user's preference for an item to be positive.

We expect that TC-MF can better differentiate between positive and negative feedback because of this modeling choice.

With f_L, $\mathcal{F}$, and δ as defined previously, we consider the following likelihood function:

$$p^{LK} = \prod_{(u,i)\in M\times N} p(r^+_{u,i})^{\delta(f_L(u,i)=+1)}(1-p(r^+_{u,i}))^{\delta(f_L(u,i)=-1)} \tag{5}$$

And the resulting log-likelihood function, when expanded is:

$$\log p^{LK} = \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\mathcal{F} - \Big(\delta(f_L(u,i)=+1) + \delta(f_L(u,i)=-1)\Big)\log(1+\exp(\mathcal{F})) \tag{6}$$

For both TC-MF1 and TC-MF, following (6), we aim to maximize the following objective function, with additional L-2 regularization terms:

$$\max_{a,b,\beta_u,\beta_i} \sum_{(u,i)\in M\times N} \log p^{LK} - \frac{\lambda}{2}\|a\|_2 - \frac{\lambda}{2}\|b\|_2 \tag{7}$$

4.3 S-MF

The above models, being extensions of the One-Class Collaborative Filtering models, can only differentiate positive links from negative links by estimating a single probability term $p(r^+_{ui})$. Intuitively, capturing the signals in the unknown links (?), together with positive (+) and negative (−) links should provide a richer model. To better model the Two-Class Collaborative Filtering problem, we assume that each item has a positive latent factor and another negative latent factor. This is shown as a graphical model in Fig. 1. Each

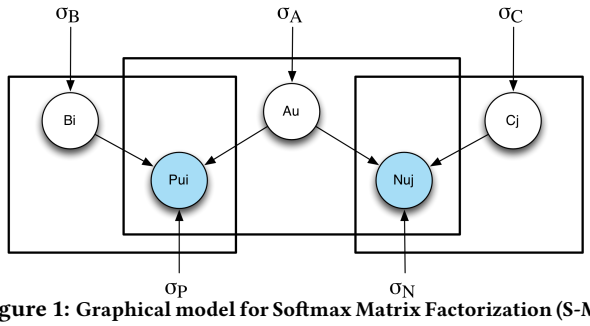

Figure 1: Graphical model for Softmax Matrix Factorization (S-MF)

user has a single latent factor that interacts with both latent factors of each item. This model assumes each user-item interaction to be part of one of the three classes: +, −, or 0. Modeling the problem this way, we can estimate the probabilities for a positive or negative preference towards an item.

We will learn a combination of two independent logistic regressions, representing positive (P) and negative (N) item membership with common latent user features $\mathbf{A}$, as well as positive ($\mathbf{B}$) and negative latent item features ($\mathbf{C}$).

$$\text{Positive (P): } \hat{\mathbf{R}}^+ = \mathbf{A}\mathbf{B}^\mathbf{T} + \beta^+_\mathbf{i} + \beta_\mathbf{u}$$

$$\text{Negative (N): } \hat{\mathbf{R}}^- = \mathbf{A}\mathbf{C}^\mathbf{T} + \beta^-_\mathbf{i} + \beta_\mathbf{u}$$

We denote the probability that user u likes item i as:

$$p(r^+_{ui} \mid a_u, b_i, c_i, \beta^+_i, \beta^-_i, \beta_u) = \frac{\exp(a_u b_i^T + \beta^+_i + \beta_u)}{1+\exp(a_u b_i^T + \beta^+_i + \beta_u) + \exp(a_u c_i^T + \beta^-_i + \beta_u)}$$

And the probability that user u dislikes item k as:

$$p(r^-_{uk} \mid a_u, b_k, c_k, \beta^+_k, \beta^-_k, \beta_u) = \frac{\exp(a_u c_k^T + \beta^-_j + \beta_u)}{1+\exp(a_u b_k^T + \beta^+_k + \beta_u) + \exp(a_u c_k^T + \beta^-_k + \beta_u)}$$

And therefore the probability that it is unknown if the user u likes or dislikes the item j as:

$$p(r^0_{uj} \mid a_u, b_j, c_j, \beta^+_j, \beta^-_j, \beta_u) = 1 - p(r^+_{uj} \mid a_u, b_j, c_j, \beta^+_j, \beta^-_j, \beta_u) - p(r^-_{uj} \mid a_u, b_j, c_j, \beta^+_j, \beta^-_j, \beta_u)$$

Again, for simplicity we omit the conditionals and denote the probabilities as $p(r^+_{ui})$, $p(r^-_{ui})$, and $p(r^0_{ui})$. Given the following likelihood function with symbols as defined previously:

$$P^{LK} = \prod_{(u,i)\in M\times N} p(r^+_{ui})^{\delta(f_L(u,i)=+1)} p(r^-_{ui})^{\delta(f_L(u,i)=-1)} p(r^0_{ui})^{\delta(f_L(u,i)=0)}$$

we get this logistic likelihood function which we want to maximize:

$$\log P^{LK} = \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)(a_u b_i^T + \beta^+_i + \beta_u) + \delta(f_L(u,i)=-1)(a_u c_i^T + \beta^-_i + \beta_u) - \log\left(1 + e^{a_u b_i^T + \beta^+_i + \beta_u} + e^{a_u c_i^T + \beta^-_i + \beta_u}\right)$$

Optimization. In order to maximize the logistic likelihood functions for all three methods described above, we first find their partial derivatives in terms of the user and item latent factors. We use the gradients to update the user and item factors iteratively. We

	M/N	Edges	Density	+Links
Slashdot	7,896/9,504	305,583	0.004	75.74%
WikiVote	6,129/2,384	103,185	0.007	76.89%
MovieLens	6,040/3,706	998,087	0.044	54.56%
Yelp	6,945/11,274	316,162	0.004	57.64%
BookCrossing	3,398/14,841	306,669	0.006	28.78%

Table 1: Statistics of the evaluated datasets.

begin with random factors and at each step we keep the user factors constant while updating the item factors. In the next step we keep the item factors constant while updating the user factors. This alternating gradient approach is used until convergence or until a certain number of iterations have been made.

The gradients of user and item factors can be derived from the expressions above; we include them in a supplementary document for space reasons[1].

5 EXPERIMENTS AND EVALUATION

In this section we provide a detailed description of our evaluation scheme. We want to investigate the following: (a) overall (listwise) ranking accuracy and ranking quality at two ends of the list (top-k and bottom-k), which are necessary for many applications, (b) recommendation diversity of different models, (c) behavior of different models with the change in model parameter, viz. number of latent factors k, and regularization constant λ, (d) performance of the more complex S-MF model as more training data is available. Recall that S-MF has twice as many parameters for each item. More details about our evaluation is available at https://github.com/uzh/tccf.

5.1 Setup

First we introduce the datasets, evaluation measures, and baseline methods we compare with.

Datasets and Train/Test Splits. We evaluate our models on five well-known and open datasets: Slashdot [12], WikiVote [3], MovieLens [2], BookCrossing [20], and Yelp [3] (restaurants only). The first two are signed directed social networks and the others are from the recommender systems domain. In order to have sufficient training and test data, we make sure that each node has received and given a certain number of feedbacks.

The Slashdot dataset contains feedback of users as Friends (positive link) or Foes (negative link) by other users. WikiVote comprises information about users voting for (positive link) or against (negative link) admin candidates in the Wikipedia community voting.

Unlike Slashdot and WikiVote, other datasets do not have explicit negative feedback, but user preference is expressed as a numeric rating value (e.g., between 1 and 5). To convert numeric ratings into positive and negative feedback, we subtract each rating of a user by the mean rating provided by the user over all items. An alternative is to consider high ratings (e.g., 4-5) as positive and low ratings (1-2) as negative. However, some users may have a high rating bias and only provide ratings between 3 and 5. By applying mean-removal from each user's ratings, we address this problem and convert numeric ratings to user specific positive or negative preference over items.

The properties of the resulting datasets are described in Table 1. We can see that MovieLens is the least sparse dataset and all datasets other than BookCrossing have a higher proportion of positive feedbacks. The frequency distribution of positive and negative links is shown in Figure 2. The distribution is long tailed for both positive and negative links, but towards the tail all datasets other than BookCrossing have even fewer negative links.

For evaluations, we create five different test and train splits of each dataset. Each test split is generated by randomly sampling 30% of the total feedbacks such that in the resulting training split: (a) every node (user) has given at least five positive and three negative feedbacks, and (b) every nodes (item) has received at least one positive and one negative feedback. We train the models on the training sets and generate rankings. The models then are evaluated based on how many links in the test set are successfully predicted at the relevant ends (top or bottom) of the ranked lists. All evaluations are performed by doing 5-fold cross validation over these splits and taking average value for each measurement.

Evaluation Measures. Apart from AUC and GAUC measures introduced in Section 2, we use other common measures to evaluate the performance of our models. **AUC** and **GAUC** describe the quality of the entire ranking list by quantifying how better it is from a random ordering. Although they are useful measures to compare different models, for most applications only a small subset of the ranking is ever used. User experience with search engines, social networks or recommender systems involves interaction with such small personalized rankings, usually including 10-20 suggestions.

Our goal is to generate a ranking of links such that more positive links and few negative links (flops) feature at the top of the list. In other words, we want to simultaneously have two different types of links at the two ends of the ranked list.

A *positive hit* (or true positive) and a *negative hit/flop* (or true negative) are a positive- and a negative link from the test set, respectively. Precision at top-k (**P@k**) counts the number of hits among the top-k links of the ranked list divided by the cut-off level k. Similarly, Precision at bottom-k (**P@-10**) is the number of negative hits in the bottom-k divided by k. Hit Rate at top-k (**H@k**) is the total number of positive hits predicted by the model at top-k divided by the number of users. Likewise, Hit Rate at bottom-k (**H@-k**) is the total number of negative hits at bottom-k divided by the number of users.

Apart from having many positive links at the top, it is also desirable to have fewer flops at the top. We measure this quality using **NI@k**, which is calculated as $\frac{\#\text{negative links at top-k}}{k}$.

All the above measures are concerned with the accuracy of ranking. A good recommendation system is one that additionally provides the users with diverse suggestions. Item Coverage at top-k (**IC@k**) measures the total number of unique suggestions on the top-k recommendation for all users divided by the total number of recommendable links. Similarly, we measure the average popularity of recommended links at top-k by **Id@k** (Item-degree). A very high Id@k means that mostly well-known suggestions are made to the users. These *blockbuster* suggestions are probably already known to the users. Therefore, a very high Id@k is not a desirable quality. A more detailed discussion of recommendation diversity and diversity measures can be found in [1, 4, 14].

The values of P@k, H@k, AUC, GAUC and NI@k reported in this section are averaged over all users. Higher values P@k, H@k, AUC, and GAUC indicate better ranking, while a lower value of

[1]Supplementary document http://www.merlin.uzh.ch/publication/show/15001
[2]http://grouplens.org/datasets/movielens/1m/
[3]http://www.ics.uci.edu/~vpsaini/

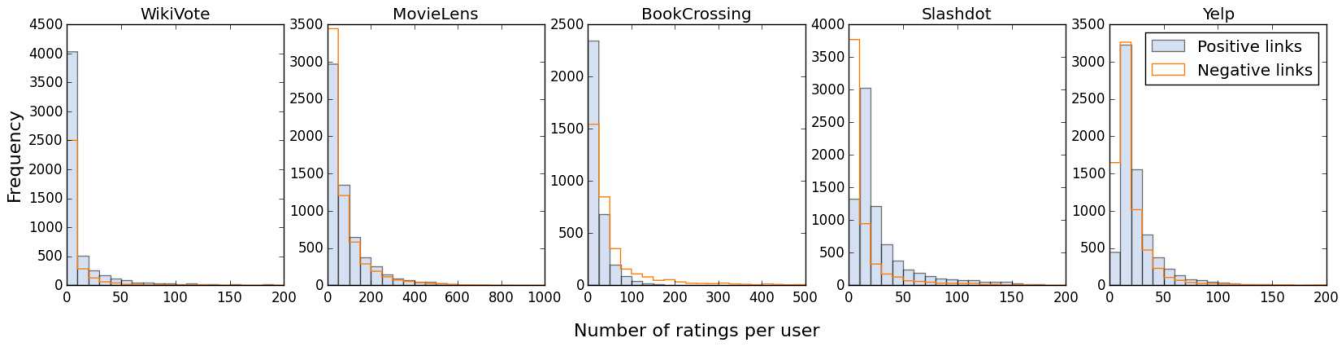

Figure 2: Distribution of different types of links per user. The horizontal axis shows the number of links per user and the vertical axis shows the frequency. As an example, for WikiVote, the number of users with 0-10 negative links is about 2,500.

NI@k is desirable. For IC@k, a higher value is desirable, while for Id@k a lower value is desirable.

Baselines and Parameters. We compare our methods with BPRMF [15], GAUC-OPT [19], and Matrix Factorization [18] model with logistic loss [10]. All of these methods have the same parameter k, which is the number of latent factors. Similarly, they all use the regularization constant λ and BPRMF has additional regularizer for negative item factors, which is $\lambda/10$. In order to be fair to the compared systems, we evaluated with parameters similar to [19], as well as some additional ones. We ran experiments with $\lambda \in \{0.001, 0.01, 1, 5, 10, 20, 50, 100\}$. Similarly, we varied the number of factors within $k \in \{10, 30, 50, 70, 90\}$. We also varied learning rates between 0.001 and 0.01. For TC-MF1, TC-MF, S-MF and MF, we used AdaGrad [5] with a single parameter of $\gamma = 1$ instead of using different learning rates. We want to see if our models performs well without searching on a lot of different parameter settings. We ran all models until convergence or a maximum of 30 iterations.

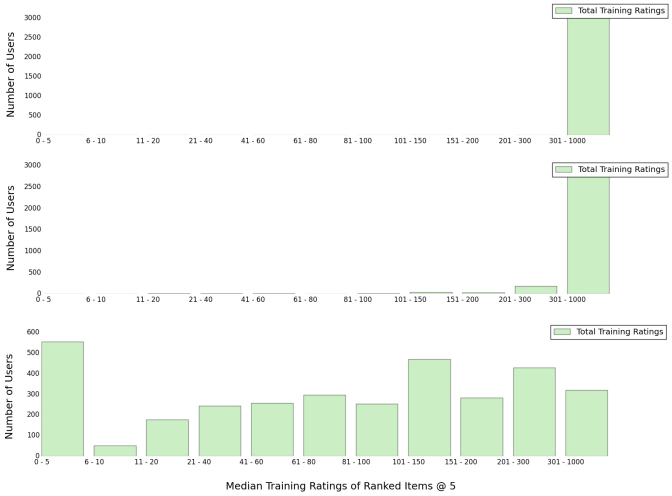

Figure 3: Median Item Degree at top-k (Id@5) on Slashdot dataset. From top to bottom: BPRMF, TC-MF, and S-MF. S-MF recommends links from all bands while others recommend many popular links.

5.2 Result and Discussion

We now present the result of our experimental evaluations and compare the performance of our methods with the state-of-the-art methods.

Model	AUC	GAUC	P@ 5	P@ 10	P@ -5	P@ -10	NI@ 5	H@ 5	H@ 10	IC@ 5	Id@ 5
Slashdot											
TC-MF	**0.77**	**0.68**	**0.09**	**0.07**	**0.029**	**0.027**	**0.002**	**0.30**	**0.41**	0.027	580.34
TC-MF1	0.60	0.59	0.07	0.06	0.001	0.001	0.005	0.25	0.36	0.025	662.79
S-MF	0.58	0.56	0.01	0.01	0.005	0.002	0.004	0.07	0.10	**0.236**	**137.4**
MF	0.68	0.63	0.07	0.06	0.023	0.019	0.004	0.27	0.37	0.019	786.08
BPRMF	0.61	0.60	0.08	0.06	0.001	0.001	0.004	0.29	0.40	0.001	801.67
GAUC-OPT	0.68	0.62	0.05	0.05	0.013	0.012	0.003	0.18	0.27	0.211	163.73
WikiVote											
TC-MF	0.65	0.64	0.02	0.03	**0.016**	**0.016**	0.008	0.10	0.18	0.06	60.29
TC-MF1	0.64	0.64	0.03	0.03	0.001	0.001	0.008	0.12	0.24	**0.4**	83.40
S-MF	0.69	0.67	0.04	0.04	0.013	0.012	0.009	0.17	0.26	0.369	61.56
MF	**0.74**	**0.69**	0.02	0.03	0.004	0.004	**0.003**	0.10	0.19	0.299	**57.6**
BPRMF	0.62	0.63	**0.07**	**0.06**	0.001	0.001	0.023	**0.27**	**0.39**	0.005	148.69
GAUC-OPT	0.66	0.62	0.04	0.04	0.007	0.008	**0.003**	0.16	0.28	0.398	78.29
MovieLens											
TC-MF	0.74	0.64	0.17	0.16	0.032	0.030	0.019	0.49	0.63	0.072	1439.26
TC-MF1	0.65	0.58	0.18	0.17	0.001	0.001	0.038	0.51	0.66	0.010	1861.91
S-MF	0.82	0.81	0.13	0.12	**0.041**	**0.043**	0.038	0.43	0.56	**0.442**	967.83
MF	**0.84**	**0.83**	0.07	0.07	0.005	0.006	0.016	0.25	0.39	0.256	**740.72**
BPRMF	0.68	0.58	**0.21**	**0.20**	0.000	0.000	0.036	**0.60**	**0.75**	0.051	1442.08
GAUC-OPT	0.69	0.62	0.14	0.13	**0.041**	0.041	**0.015**	0.55	0.55	0.335	882.58
Yelp											
TC-MF	0.64	0.60	**0.01**	**0.01**	**0.005**	**0.004**	0.003	0.05	0.10	0.0009	238.3
TC-MF1	0.59	0.57	**0.01**	**0.01**	0.000	0.000	0.004	0.05	0.09	0.041	236.42
S-MF	0.57	0.56	**0.01**	**0.01**	0.004	**0.004**	0.004	0.04	0.07	**0.39**	55.56
MF	**0.75**	**0.72**	0.00	0.00	0.000	0.000	**0.001**	0.02	0.03	0.309	**47.99**
BPRMF	0.60	0.58	**0.01**	**0.01**	0.000	0.000	0.005	**0.07**	**0.12**	0.002	209.29
GAUC-OPT	0.54	0.53	**0.01**	**0.01**	0.002	0.002	0.003	0.04	0.08	0.29	74.51
BookCrossing											
TC-MF	0.52	0.56	0.01	0.01	**0.033**	**0.028**	0.003	0.03	0.04	0.0007	74.93
TC-MF1	0.50	0.55	0.00	0.00	0.006	0.006	0.005	0.01	0.02	**0.312**	32.35
S-MF	0.55	0.58	0.00	0.00	0.010	0.007	0.008	0.02	0.03	0.19	55.56
MF	**0.70**	**0.71**	0.00	0.00	0.001	0.001	**0.001**	0.00	0.00	0.068	**6.2**
BPRMF	0.50	0.42	**0.02**	**0.02**	0.000	0.001	0.024	**0.08**	**0.14**	0.001	92.37
GAUC-OPT	0.52	0.52	0.00	0.00	0.013	0.013	0.003	0.02	0.03	0.235	42.63

Table 2: Performance for parameters corresponding to the maximum value of GAUC. Measures above the dotted line indicate the performance of our models for each dataset.

Listwise and Top-k Ranking. We present the result of all evaluated methods for their parameter values corresponding to maximum GAUC in Table 2. The best measure in each column is bold-faced. From Table 2, we can see that TC-MF and S-MF are often better than BPRMF and GAUC-OPT in terms of list-wise measures (AUC and GAUC). While MF outperforms our models in listwise measures, it performs poorly in the top-k measure at the top (P@5, P@10, H@5, H@10) and bottom (P@-5, P@-10) of the list. Although BPRMF performs better at ranking positive items at the top of the list, we can see that TC-MF has much lower proportion of flops at the top of the list (NI@5). For example, in case of WikiVote and BookCrossing datasets, BPRMF has about three to eight times more flops at the top of the list compared to TC-MF, and two to three times more compared to S-MF. Recall that the BookCrossing dataset has more

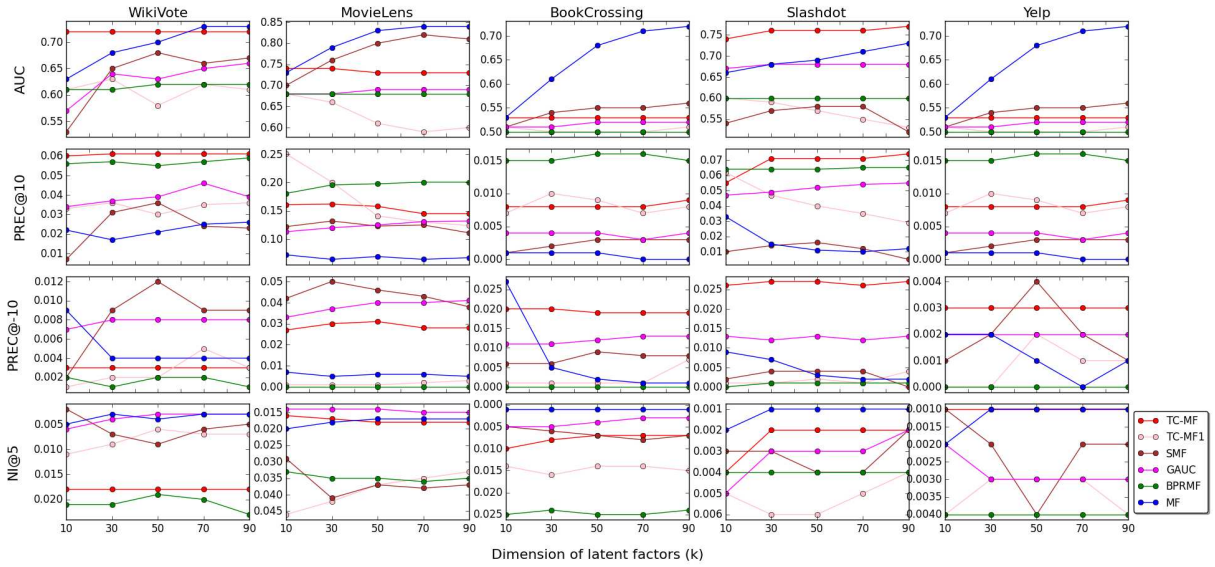

Figure 4: AUC, Prec@10, Prec@-10, and NI@5 at different values of k (dimension of latent factors). Note that the y-axis for NI@5 has been inversed, so that the highest curve indicates the best performance.

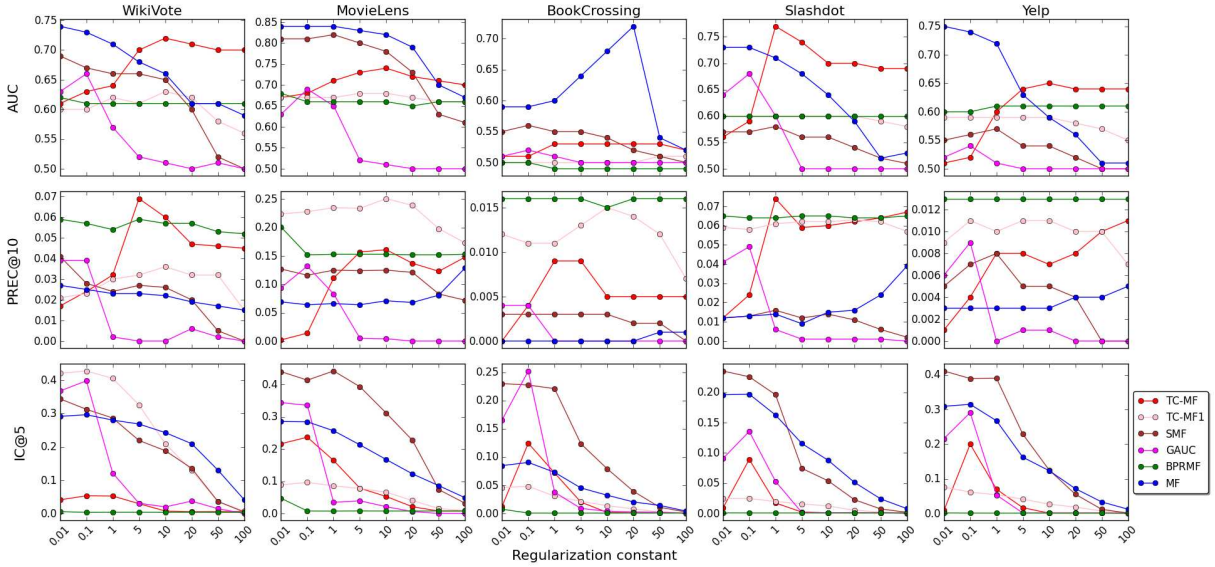

Figure 5: AUC, Prec@10, and IC@5 at different values of λ (regularization constant).

negative links than positive ones. Similarly, while GAUC-OPT is good at ranking negative items at the bottom, it is not as good in distinguishing positive items. On datasets other than MovieLens, S-MF also has fewer flops at the top of the list (NI@5) compared to BPRMF.

For ranking negative links at the bottom of the list (P@-5, P@-10), our models do better than others for all datasets. Similarly, in case of the very sparse Slashdot dataset, TC-MF outperforms every other model. For the remaining datasets, they are among the top or second-top performing model across both positive and negative ranking measures towards the top and bottom of the ranking list. We also observe TC-MF usually performs better than S-MF. As S-MF is more complex, we expect it to model the problem better and generate superior ranking than TC-MF. However, since S-MF also has to learn a lot more parameters than TC-MF, the sparsity of training data might affect the former's performance. We investigate this issue later in this section. From the above discussion, we can suggest that our models are suitable for a general ranking task, as well as the task of TCCF at both ends of the list, i.e., more positive items and fewer flops at the top of the list.

Recommendation Diversity. As described in Section 5.1, we measure recommendation diversity in terms of Item Coverage (IC@5) and median degree of recommended links in the top-k list (Id@5). A lower value of IC@5 is not desirable as it means that few common links appear in the top-5 ranking for many users. Likewise, a higher value of Id@5 is not desirable since it suggests that very-popular links are being suggested to the users, which might suggest a boring recommendation of blockbuster items.

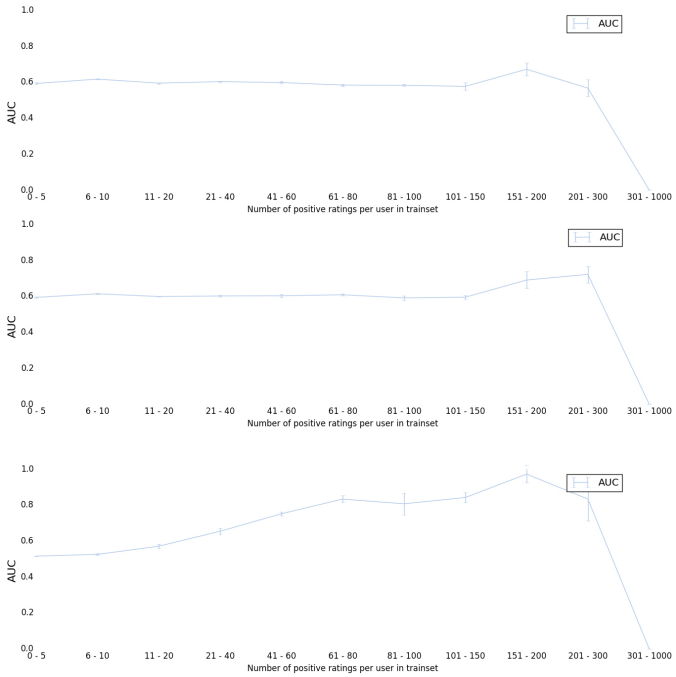

Figure 6: AUC for users with varying amount of training data on the Slashdot dataset, from top to bottom: BPRMF, TC-MF, and S-MF.

From Table 2, we see that BPRMF has very low IC@5 values for all datasets, while S-MF and TC-MF recommend more low-degree links and are better in IC@5 measure. Although BPRMF performs better in P@5 and H@5, it seems to achieve this by recommending few popular links to all users, which we can observe in the Id@5 column. In Figure 3, we show the frequency distribution of Id@5 for BPRMF, TC-MF, and S-MF on the Slashdot dataset. While S-MF recommends items from all bands, other methods recommend mostly popular items only. This behavior holds across other datasets too, but we omit them here for space reasons. Thus, we can see that our model produces more diverse recommendations.

Variation with k and λ. We plot the variation of four different measures with the model parameter (dimension of latent features, k) in Figure 4, by choosing the best performing regularization constant λ for each model. Notice that the y-axis for NI@5 is inverted, which means that a higher curve indicates better performance.

From Figure 4, we see that at $k = 10$, TC-MF has the highest AUC on all datasets and is stable across different values of k. The advantage of using fewer dimensions is cheaper storage and more compressed representation of the dataset, allowing quick computation of recommendation lists. The other interpretation is that TC-MF is not able to exploit higher dimension features to capture more nuances in the dataset. In other words, it indicates that TC-MF is a simple model with the ability to generalize well with few parameters, i.e., without over-fitting. The AUC values of S-MF and MF improve as k increases.

For P@10, the best models across multiple k values are BPRMF, TC-MF1 and TC-MF. Likewise, for P@-10, the best models are SMF, GAUC-OPT, and TC-MF. BPRMF has consistently low values for NI@5, indicating higher proportion of flops at the top of the list.

In Figure 5, we choose the best value of k for each model and plot the variation of three different measures with the regularization parameter λ. The most revealing finding from Figure 5 is that as the regularization parameter λ increases, Item Coverage IC@5 decreases and approaches 0. Most previous work using matrix factorization methods only report their best performance using any regularization parameter, but this finding suggests that the choice of regularization parameter is an important consideration. A limited set of regularization parameters might produce a better recommendation in terms of some measure like AUC, while at the same time being very limited in terms of diversity. For this reason we evaluate with $\lambda \in \{0.001, 0.01, 1, 5, 10, 20, 50, 100\}$, while regularization parameters only above 1 are reported in [19] . In Figure 5, we see that a higher AUC value is possible at the cost of a low recommendation diversity. We can also see that S-MF has higher diversity as well as higher accuracy (AUC) and modest top-k performance (P@10) at lower values of λ. On the other hand, BPRMF has higher accuracy values, but at the cost of diversity.

Performance of S-MF with more training data. As we discussed in the above sections, while S-MF does good in some measures on some datasets, it still underperforms other models despite being a more complex model. S-MF needs to learn twice as many parameters for each item compared to the other models. We would like to investigate if lack of training data because of the skewed rating distribution explains this difference in performance. In Figure 6, we present the AUC values of three top-performing models BPRMF, TC-MF, and SMF on the Slashdot dataset for users with varying number of positive links in the training set.

We can observe from this figure that the AUC performance for other models remains relatively flat throughout, while it gradually increases and approaches 1.0 for S-MF as more training data is available. This behavior is also observed in other datasets. This suggests that S-MF is able to produce a better ranking as more training data is available.

6 CONCLUSION

In this paper, we described the Two Class Collaborative Filtering Problem. Ranking positive items at the top of a ranking list is important, but so is being able to distinguish negative items (flops) and rank fewer of them at the top. This problem has only recently received attention because of the increased availability of negative feedback and research on signed social networks. A related problem is that of producing a ranking list with higher diversity, i.e., of links that are not already well-known. In other words, it is important for recommendation systems to generate rankings that are both accurate on the one hand, and not boring on the other hand.

We introduced new probabilistic latent factor models that are able to perform well on the above defined TCCF problem. Specifically, TC-MF model is an extension of the esiting MF model that is simple, yet more accurate in the TCCF task than existing models. The more complex S-MF model produces better rankings but it needs more training data to do so. Our models also produce more diverse and non-blockbuster recommendations.

In the future, we're interested to look into improving the predictive performance of these model. We also want to incorporate side information to make them more robust and perform better with fewer training data.

REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. 2012. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2012), 896–911.

[2] John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 43–52.

[3] Moira Burke and Robert Kraut. 2008. Mopping up: modeling wikipedia promotion decisions. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*. ACM, 27–36.

[4] Pablo Castells, Neil J Hurley, and Saul Vargas. 2015. Novelty and diversity in recommender systems. In *Recommender Systems Handbook*. Springer, 881–918.

[5] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[6] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (1992), 61–70.

[7] James A Hanley and Barbara J McNeil. 1983. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology* 148, 3 (1983), 839–843.

[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[9] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

[10] Christopher C Johnson. 2014. Logistic matrix factorization for implicit feedback data. In *NIPS 2014 Workshop on Distributed Machine Learning and Matrix Computations*.

[11] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[12] Cliff AC Lampe, Erik Johnston, and Paul Resnick. 2007. Follow the reader: filtering comments on slashdot. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1253–1262.

[13] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 502–511.

[14] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. 2016. Updatable, Accurate, Diverse, and Scalable Recommendations for Interactive Applications. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 1 (Dec. 2016), 34 pages. DOI: http://dx.doi.org/10.1145/2955101

[15] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[16] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 175–186.

[17] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. ACM, 880–887.

[18] Ruslan Salakhutdinov and Andriy Mnih. 2011. Probabilistic matrix factorization. Citeseer.

[19] Dongjin Song and David A Meyer. 2015. Recommending Positive Links in Signed Social Networks by Optimizing a Generalized AUC.. In *AAAI*. 290–296.

[20] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.

7 SUPPLEMENTARY MATERIAL

Here we present the material not contained in the main text due to lack of space. First, we present the gradients of expressions from Section 4, followed by eperimental results that supplement those in Section 5.

The notations used in this section have been described in the main text: f_L in Section 2, $\mathcal{F}$ in Section 4.1, and δ is an indicator function. The expressions for log-likelihood functions have also been described alongside the relevant models in Section 4.

7.1 Gradients for TC-MF1

We aim to maximize the following objective function for both TC-MF1 and TC-MF

$$\max_{a,b,\beta_u,\beta_i} \sum_{(u,i)\in M\times N} \log p^{LK} - \frac{\lambda}{2}\|a\|_2 - \frac{\lambda}{2}\|b\|_2 \tag{S1}$$

,

where the log-likelihood function is:

$$\log p^{LK} = \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\mathcal{F} - \log(1+\exp(\mathcal{F})) \tag{S2}$$

The expanded form of the log-likelihood function (S2) is given below:

$$\begin{aligned}
\log p^{LK} &= \sum_{(u,i)\in M\times N} \log\left(\frac{\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})}^{\delta(f_L(u,i)=+1)} \frac{1}{1+\exp(\mathcal{F})}}{\frac{1}{1+\exp(\mathcal{F})}^{\delta(f_L(u,i)=+1)}}\right) \\
&= \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\log\left(\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})}\right) \\
&\quad + \log\left(\frac{1}{1+\exp(\mathcal{F})}\right) \\
&\quad - \delta(f_L(u,i)=+1)\log\left(\frac{1}{1+\exp(\mathcal{F})}\right) \\
&= \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\mathcal{F} - \log(1+\exp(\mathcal{F}))
\end{aligned} \tag{S3}$$

From the above expressions, the partial derivatives for the user vectors and bias can be obtained as:

$$\begin{aligned}
\frac{\partial}{\partial a} &= \delta(f_L(u,i)=+1)b - b\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})} - \lambda a \\
\frac{\partial}{\partial \beta_u} &= \delta(f_L(u,i)=+1) - \frac{\exp(\mathcal{F}}{1+\exp(\mathcal{F})}
\end{aligned} \tag{S4}$$

Similarly for the item vectors:

$$\frac{\partial}{\partial b} = \delta(f_L(u,i)=+1)a - a\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})} - \lambda b \tag{S5}$$

7.2 Gradients for TC-MF

We aim to maximize the objective function similar to (S1), where the log-likelihood function is:

$$\begin{aligned}
\log p^{LK} &= \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\mathcal{F} \\
&\quad - \Big(\delta(f_L(u,i)=+1) + \delta(f_L(u,i)=-1)\Big)\log(1+\exp(\mathcal{F}))
\end{aligned} \tag{S6}$$

The expanded form of the log-likelihood function (S6) is given below:

$$\begin{aligned}
\log p^{LK} &= \sum_{(u,i)\in M\times N} \log\Big(\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})}^{\delta(f_L(u,i)=+1)} \\
&\qquad \frac{1}{1+\exp(\mathcal{F})}^{\delta(f_L(u,i)=-1)}\Big) \\
&= \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\log\left(\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})}\right) + \\
&\qquad \delta(f_L(u,i)=-1)\log\left(\frac{1}{1+\exp(\mathcal{F})}\right) \\
&= \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)\mathcal{F} - (\delta(f_L(u,i)=+1) + \\
&\qquad \delta(f_L(u,i)=-1)\log(1+\exp(\mathcal{F}))
\end{aligned} \tag{S7}$$

From the above expressions, the partial derivatives for the user vectors and bias can be obtained as:

$$\begin{aligned}
\frac{\partial}{\partial a} &= \delta(f_L(u,i)=+1)b - (\delta(f_L(u,i)=+1) + \\
&\qquad \delta(f_L(u,i)=-1))b\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})} - \lambda a \\
\frac{\partial}{\partial \beta_u} &= \delta(f_L(u,i)=+1) - (\delta(f_L(u,i)=+1) + \\
&\qquad \delta(f_L(u,i)=-1)\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})}
\end{aligned} \tag{S8}$$

Similarly for the item vectors:

$$\begin{aligned}
\frac{\partial}{\partial b} &= \delta(f_L(u,i)=+1)a - (\delta(f_L(u,i)=+1) + \\
&\qquad \delta(f_L(u,i)=-1))a\frac{\exp(\mathcal{F})}{1+\exp(\mathcal{F})} - \lambda b
\end{aligned} \tag{S9}$$

7.3 Gradients for S-MF

We want to maximize the objective function:

$$\max_{a,b,c,\beta_u,\beta_i^+,\beta_i^-} \sum_{(u,i)\in M\times N} \log p^{LK} - \frac{\lambda}{2}\|a\|_2 - \frac{\lambda}{2}\|b\|_2 - \frac{\lambda}{2}\|c\|_2 \tag{S10}$$

,

where the log-likelihood function is:

$$\begin{aligned}
\log P^{LK} &= \sum_{(u,i)\in M\times N} \delta(f_L(u,i)=+1)(a_u b_i^T + \beta_i^+ + \beta_u) \\
&\qquad + \delta(f_L(u,i)=-1)(a_u c_i^T + \beta_i^- + \beta_u) \\
&\qquad - \log\left(1 + e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}\right)
\end{aligned} \tag{S11}$$

From the above expressions, the partial derivatives for different user and item factors, and biases can be obtained as given below.

$$\frac{\partial}{\partial a_u} = \begin{cases} b_i - (b_i e^{a_u b_i^T + \beta_i^+ + \beta_u} + c_i e^{a_u c_i^T + \beta_i^- + \beta_u}) \frac{1}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{if } f_L(u,i) = +1, \\ c_i - (b_i e^{a_u b_i^T + \beta_i^+ + \beta_u} + c_i e^{a_u c_i^T + \beta_i^- + \beta_u}) \frac{1}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{if } f_L(u,i) = -1, \\ -(b_i e^{a_u b_i^T + \beta_i^+ + \beta_u} + c_i e^{a_u c_i^T + \beta_i^- + \beta_u}) \frac{1}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial b_i} = \begin{cases} a_u - a_u \frac{e^{a_u b_i^T + \beta_i^+ + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{if } f_L(u,i) = +1, \\ -a_u \frac{e^{a_u b_i^T + \beta_i^+ + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial c_i} = \begin{cases} a_u - a_u \frac{e^{a_u c_i^T + \beta_i^- + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{if } f_L(u,i) = -1, \\ -a_u \frac{e^{a_u c_i^T + \beta_i^- + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial \beta_u} = \begin{cases} 1 - \frac{e^{a_u b_i^T + \beta_i^+ + \beta_u +} + e^{a_u c_i^T + \beta_i^- + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{if } f_L(u,i) = +1 \text{ or } f_L(u,i) = -1, \\ -\frac{e^{a_u b_i^T + \beta_i^+ + \beta_u +} + e^{a_u c_i^T + \beta_i^- + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial \beta_i^+} = \begin{cases} 1 - \frac{e^{a_u b_i^T + \beta_i^+ + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{if } f_L(u,i) = +1, \\ -\frac{e^{a_u b_i^T + \beta_i^+ + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{else} \end{cases}$$

$$\frac{\partial}{\partial \beta_i^-} = \begin{cases} 1 - \frac{e^{a_u c_i^T + \beta_i^- + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{if } f_L(u,i) = -1, \\ -\frac{e^{a_u c_i^T + \beta_i^- + \beta_u}}{1+e^{a_u b_i^T + \beta_i^+ + \beta_u} + e^{a_u c_i^T + \beta_i^- + \beta_u}} & \text{else} \end{cases}$$

(S12)